

ISIS XI PL/M-86 V2.0 COMPILATION OF MODULE HELP

OBJECT MODULE PLACED IN HELP.OBJ

COMPILER INVOKED BY: IF0: HELP.PLN DEBUG XREF OPTIMIZE(2) PAGESWIDTH(100)

```

1      $title ('Help Utility Version 1.1')
      help:
      do;

/*
      Copyright (C) 1982
      Digital Research
      P.O. 579
      Pacific Grove, CA 93950

      Revised:
      8 Jan 82 by Bruce Skidmore
*/

2 1      declare plm label public;

/******
      Interface Procedures
      *****/

3 1      mon1:
          procedure (func,info) external;
4 2          declare func byte;
5 2          declare info address;
6 2          end mon1;

7 1      mon2:
          procedure (func,info) byte external;
8 2          declare func byte;
9 2          declare info address;
10 2         end mon2;

```

CP/M-86 v.1.1 (Vol. II)
 COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950
 SER. # C81-162

```

/******
      Global Variables
      *****/

11 1      declare (print$mode,create$mode,extract$mode) byte;
12 1      declare (offset,eod) byte;

13 1      declare fcb (13) byte external;
14 1      declare fcb2 (36) byte;

15 1      declare maxb address external;
16 1      declare fcb16 (1) byte external;
17 1      declare tbuff (128) byte external;

18 1      declare control$z literally '1AH';
19 1      declare cr literally 'ODH';
20 1      declare lf literally 'OAH';
21 1      declare tab literally '09H';
22 1      declare slash literally '///';
23 1      declare true literally 'OFFH';

```



```
53 2      procedure (fcb$address);
54 2          declare fcb$address address;
55 2          call mon1(19,fcb$address);
56 2      end delete$file;

56 1      open$file:
57 2          procedure (fcb$address) byte;
58 2          declare fcb$address address;
59 2          return mon2 (15,fcb$address);
60 2      end open$file;

60 1      close$file:
61 2          procedure (fcb$address) byte;
62 2          declare fcb$address address;
63 2          return mon2 (16,fcb$address);
64 2      end close$file;

64 1      read$record:
65 2          procedure (fcb$address) byte;
66 2          declare fcb$address address;
67 2          return mon2 (20,fcb$address);
68 2      end read$record;

68 1      write$record:
69 2          procedure (fcb$address) byte;
70 2          declare fcb$address address;
71 2          return mon2(21,fcb$address);
72 2      end write$record;

72 1      make$file:
73 2          procedure (fcb$address) byte;
74 2          declare fcb$address address;
75 2          return mon2(22,fcb$address);
76 2      end make$file;

76 1      read$rand:
77 2          procedure (fcb$address) byte;
78 2          declare fcb$address address;
79 2          return mon2(33,fcb$address);
80 2      end read$rand;

80 1      set$dma:
81 2          procedure (dma$address);
82 2          declare dma$address address;
83 2          call mon1(26,dma$address);
84 2      end set$dma;

84 1      set$rand$rec:
85 2          procedure (fcb$address);
86 2          declare fcb$address address;
87 2          call mon1(36,fcb$address);
88 2      end set$rand$rec;

88 1      terminate:
89 2          procedure;
90 2          call mon1 (0,0);
91 2      end terminate;
```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. B. X 579

PACIFIC GROVE, CA 93950

SER. # _____

/*****

Move Procedure

Moves specified number of bytes
from the Source address to the
Destination address.

*****/

```

91  1  movef:
      procedure (mvcnt,source$addr,dest$addr);
92  2      declare (source$addr,dest$addr) address;
93  2      declare mvcnt byte;
94  2      call move(mvcnt,source$addr,dest$addr);
95  2      return;
96  2  end movef;

```

/*****

Compare Function

Compares 12 byte strings

Results: 0 - string1 = string2
1 - string1 < string2
2 - string1 > string2

*****/

```

97  1  compare:
      procedure(str1$addr,str2$addr) byte;
98  2      declare (str1$addr,str2$addr) address;
99  2      declare string1 based str1$addr (12) byte;
100  2      declare string2 based str2$addr (12) byte;
101  2      declare (result,i) byte;
102  2      result,
          i = 0;
103  2      do while ((i < 12) and (string1(i) <> ' '));
104  3          if string1(i) <> string2(i) then
105  3              do;
106  4                  if string1(i) < string2(i) then
107  4                      do;
108  5                          result = 1;
109  5                      end;
                      else
110  4                          do;
111  5                              result = 2;
112  5                          end;
                      i = 11;
113  4                      end;
114  4                      i = i + 1;
115  3                      end;
116  3                      return result;
117  2                      end compare;
118  2

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P.O. B X 579

PACIFIC GROVE, CA 93950

SER. # _____

/*****

Increment Procedure

Increments through a record.

*****/

```

119  1  inc:

```

```

        procedure (inci) byte;
120  2      declare inci byte;
121  2      inci = inci + 1;
122  2      if inci > 127 then
123  2          do;
124  3          if read$record(.fcb) = 0 then
125  3              do;
126  4              inci = 0;
127  4              end;
              else
128  3              do;
129  4              eod = true;
130  4              inci = 0;
131  4              end;
132  3          end;
133  2      return inci;
134  2  end inc;

```

```

/*****
Page$check Procedure

```

```

        Halts display after 23 lines
        *****/
135  1  page$check:
        procedure(line$cnt$addr) byte;
136  2      declare line$cnt$addr address;
137  2      declare line$cnt based line$cnt$addr byte;
138  2      declare quit byte;
139  2      quit = 0;
140  2      if (not print$mode) then
141  2          do;
142  3          if (line$cnt:=line$cnt+1) > 21 then
143  3              do;
144  4              call print$console$buf(.(cr,lf,'Press ENTER to continue.$'));
145  4              line$cnt = 0;
146  4              do while (line$cnt = 0);
147  5                  line$cnt = direct$con$io(OFFH);
148  5              end;
149  4              call print$console$buf(.(cr,
150  4              if line$cnt = 3 /* control c */ then
151  4                  do;
152  5                      line$cnt = close$file(.fcb);
153  5                      call terminate;
154  5                  end;
              else
155  4                  do;
156  5                      if line$cnt <> cr then
157  5                          do;
158  6                              quit = true;
159  6                              end;
160  5                              line$cnt = 0;
161  5                          end;
162  4                      end;
              else
163  3                      do;
164  4                      call write$console(lf);

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

cr,\$'));

```

165 4      end;
166 3      end;
        else
167 2      do;
168 3          line$cnt = 0;
169 3          call write$console(1f);
170 3      end;
171 2      return quit;
172 2      end page$check;

```

```

/*****

```

Init Procedure

Reads the index into memory

```

*****/

```

```

173 1      init:
        procedure;
174 2          declare (buf$size,max$buf,init$i) address;
175 2          declare end$index byte;
176 2          buf$size = maxb - .memory;
177 2          max$buf = buf$size;
178 2          end$index = 0;
179 2          init$i = 7;
180 2          do while (not end$index) and (max$buf > 127);
181 3              call set$dma(.sysbuff(init$i-7).subject);
182 3              if read$record(.fcb) <> 0 then
183 3                  do;
184 4                      init$i = close$file(.fcb);
185 4                      call print$console$buf(.(cr,lf,'Error reading HELP.HLP index',
                                                cr,lf,'$'));
186 4                      call terminate;
187 4                  end;
188 3              if sysbuff(init$i).subject(0) = '$' then end$index = true;
189 3              if not end$index then
190 3                  do;
191 4                      max$buf = max$buf - 128;
192 4                      init$i = init$i + 8;
193 4                  end;
194 3              end;
195 3          end;
196 2          call set$dma(.tbuf);
197 2          if (max$buf < 128) and (not end$index) then
198 2              do;
199 3                  call print$console$buf(.(cr,lf,'Too many entries in Index Table',
                                                cr,lf,'Not enough memory',
                                                cr,lf,'$'));
200 3                  init$i = close$file(.fcb);
201 3                  call terminate;
202 3              end;
203 2          end init;

```

```

/*****

```

Parse Procedure

Parses the command tail

```

*****/

```

```

204 1      parse:

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

procedure byte;
205 2   declare (index,cnt,i,stop,bracket) byte;
206 2   index = 0;
207 2   if tbuff(0) <> 0 then
208 2   do;
209 3       do index = 0 to 10;
210 4           call movef(15,('          ',cr,'$'),.com(index).name);
211 4       end;
212 3       do index = 1 to tbuff(0);
213 4           if tbuff(index) = tab then tbuff(index) = 20H;
214 4           else if tbuff(index) = ',' then tbuff(index) = 20H;
215 4       end;
218 3       index = 0;
219 3       cnt = 1;
220 3       stop,
221 3       bracket = 0;
222 4       do while (tbuff(cnt) <> 0) and (not stop);
223 4           if (tbuff(cnt) <> 20H) then
224 5               do;
225 5                   i = 0;
226 6                   do while (((tbuff(cnt) <> 20H) and (tbuff(cnt) <> '[')) and
227 6                       (tbuff(cnt) <> 0)) and ((i < 12) and (index < 11));
228 7                       if (tbuff(cnt) > 60H) and (tbuff(cnt) < 7BH) then
229 7                           do;
230 6                               com(index).name(i) = tbuff(cnt) - 20H;
231 7                           end;
232 7                           else
233 6                               do;
234 6                                   com(index).name(i) = tbuff(cnt);
235 6                               end;
236 5                                   cnt = cnt + 1;
237 5                                   i = i + 1;
238 5                               end;
239 6                                   index = index + 1;
240 6                                   if (bracket or (index > 10)) then
241 5                                       do;
242 5                                           stop = true;
243 6                                       end;
244 5                                           else
245 6                                               if tbuff(cnt) = '[' then
246 6                                                   do;
247 6                                                       if com(index-1).name(0) = ' ' then index = index - 1;
248 6                                                       com(index).name(0) = '[';
249 6                                                       cnt = cnt + 1;
250 6                                                       index = index + 1;
251 4                                                       bracket = true;
252 5                                                       end;
253 5                                                       else
254 4                                                           do;
255 3                                                               cnt = cnt + 1;
256 2                                                               end;
257 2                                                           end;
258 2                                                       end;
259 2                                           printmode,
260 2                                           createmode,
261 2                                           extractmode = 0;

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

257 2      if index > 0 then
258 2      do;
259 3          i = 0;
260 3          do while (i < 10);
261 4              if com(i).name(0) = 'L' then
262 4                  do;
263 5                      if (com(i+1).name(0) = 'C') then
264 5                          do;
265 6                              create$mode = true;
266 6                              index = index - 2;
267 6                          end;
268 5                      else if (com(i+1).name(0) = 'E') then
269 5                          do;
270 6                              extract$mode = true;
271 6                              index = index - 2;
272 6                          end;
273 5                      else if (com(i+1).name(0) = 'P') then
274 5                          do;
275 6                              print$mode = true;
276 6                              index = index - 2;
277 6                          end;
278 5                      else if (com(i+1).name(0) <> ' ') then
279 5                          do;
280 6                              index = index - 2;
281 6                          end;
282 5                          else
283 6                              do;
284 6                                  index = index - 1;
285 5                              end;
286 5                          i = 10;
287 4                      end;
288 4                      i = i + 1;
289 3                  end;
290 2      return index;
291 2  end parse;

```

```

/*****
Create$index Procedure

```

Creates HELP.HLP from HELP.DAT

```

*****/

```

```

292 1  create$index:
      procedure;
293 2      declare (eod, cnt, i, rec$cnt) byte;
294 2      declare (index, count, count2, max$buf, save$size) address;
295 2      declare fcb3(36) byte;
296 2      call print$console$buf(.(cr, lf, 'Creating HELP.HLP....$'));
297 2      do i = 0 to 7;
298 3          call movef(12,.( '$          '), sysbuff(i).subject);
299 3      end;
300 2      rec$cnt,
      index = 0;
301 2      save$size = maxb - .memory;
302 2      max$buf = save$size;
303 2      call movef(13,.(0, 'HELP DAT', 0), .fcb);
304 2      if open$file(.fcb) = OFFH then

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____


```

305 2      do;
306 3          call print$console$buf(.(cr,lf,'HELP.DAT not on current drive',
                                     cr,lf,'$'));
307 3          call terminate;
308 3      end;
309 2      eod = 0;
310 2      do while (not eod) and (read$record(.fcb) = 0);
311 3          i = 0;
312 3          do while(i < 128) and (not eod);
313 4              if tbuff(i) = control$z then
314 4                  do;
315 5                      eod = true;
316 5                  end;
317 4              else
318 5                  do;
319 5                      if tbuff(i) = slash then
320 6                          cnt = 0;
321 6                      do while(not eod) and (tbuff(i) = slash);
322 7                          i = inc(i);
323 7                          cnt = cnt + 1;
324 7                      end;
325 6                      if (cnt = 3) and (not eod) then
326 6                          do;
327 7                              sysbuff(index).level = tbuff(i) - '0';
328 7                              i = inc(i);
329 7                              cnt = 0;
330 7                              do while ((cnt < 12) and (not eod)) and (tbuff(i) <> cr);
331 8                                  if (tbuff(i) > 60H) and (tbuff(i) < 7BH) then
332 8                                      do;
333 9                                          sysbuff(index).subject(cnt) = tbuff(i) - 20H;
334 9                                      end;
335 8                                      else
336 9                                          do;
337 7                                              sysbuff(index).subject(cnt) = tbuff(i);
338 8                                              end;
339 8                                              i = inc(i);
340 8                                              cnt = cnt + 1;
341 7                                          end;
342 7                                          if (not eod) then
343 8                                              do;
344 8                                                  call set$rand$rec(.fcb);
345 8                                                  call movef(1,.fcb(33),sysbuff(index).record);
346 8                                                  call movef(1,.fcb(34),sysbuff(index).record+1);
347 8                                                  sysbuff(index).record = sysbuff(index).record - 0001H;
348 8                                                  sysbuff(index).rec$offset = i;
349 8                                                  index = index + 1;
350 8                                                  if ((index mod 8) = 0) then
351 9                                                      do;
352 9                                                          rect$cnt = rect$cnt + 1;
353 9                                                          max$buf = max$buf - 128;
354 9                                                          if (max$buf < 128) and (not eod) then
355 10                                         do;
                                         call print$console$buf(.(cr,lf,'Too many entries ',
                                                                    'in Index Table',
                                                                    cr,lf,'Not enough memory',
                                                                    cr,lf,'$'));

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

356 10          cnt = close$file(.fcb);
357 10          call terminate;
358 10          end;
              else
359 9            do count = index to index + 7;
360 10            call movef(12,(''
                                     .sysbuff(count).subject));

361 10          end;
362 9          end;
363 8          end;
364 7          end;
365 6          end;
              else
366 5            do;
367 6              i = inc(i);
368 6            end;
369 5          end;
370 4          end;
371 3          end;
372 2          call set$dma(.sysbuff);
373 2          rec$cnt = rec$cnt + 1;
/*****
          create HELP.HLP
*****/
374 2          call movef(13,.(0,'HELP  HLP',0),.fcb3);
375 2          call delete$file(.fcb3);
376 2          if make$file(.fcb3) = OFFH then
377 2          do;
378 3            call print$console$buf(.(cr,lf,'Unable to Make HELP.HLP',
                                     cr,lf,'$'));

379 3            cnt = close$file(.fcb2);
380 3            call delete$file(.fcb2);
381 3            cnt = close$file(.fcb);
382 3            call terminate;
383 3          end;
384 2          call movef(4,.(0,0,0,0),.fcb2+32);
385 2          cnt = read$rand(.fcb2);
386 2          fcb(32),
          fcb3(32) = 0;

387 2          do count = 0 to index - 1;
388 3            sysbuff(count).record = sysbuff(count).record + rec$cnt;
389 3          end;
390 2          do count = 0 to rec$cnt - 1;
391 3            call set$dma(.memory(shl(count,7)));
392 3            if write$record(.fcb3) = OFFH then
393 3            do;
394 4              call print$console$buf(.(cr,lf,'Write error during HELP.HLP ',
                                     'Creation',cr,lf,'$'));

395 4              cnt = close$file(.fcb3);
396 4              call delete$file(.fcb3);
397 4              cnt = close$file(.fcb2);
398 4              call delete$file(.fcb2);
399 4              cnt = close$file(.fcb);
400 4              call terminate;
401 4            end;
402 3          end;
403 2          call movef(4,.(0,0,0,0),.fcb+32);

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

404 2      cnt = read$rand(.fcb);
405 2      fcb(32) = 0;
406 2      eod = 0;
407 2      do while (not eod);
408 3          count = 0;
409 3          max$buf = save$size;
410 3          do while (not eod) and (max$buf > 127);
411 4              call set$dma(.memory(shl(count,7)));
412 4              if read$record(.fcb) <> 0 then
413 4                  do;
414 5                      eod = true;
415 5                  end;
416 4                  else
417 5                      do;
418 5                          max$buf = max$buf - 128;
419 5                          if max$buf > 128 then
420 6                              count = count + 1;
421 6                          end;
422 5                      end;
423 4                  end;
424 3          do count2 = 0 to count;
425 4              call set$dma(.memory(shl(count2,7)));
426 4              if write$record(.fcb3) = OFFH then
427 4                  do;
428 5                      call print$consoleshuf(.(cr,lf,'Write error on file HELP.HLP',
429 5                                              cr,lf,'$'));
430 5                      i = close$file(.fcb3);
431 5                      call delete$file(.fcb3);
432 5                      i = close$file(.fcb);
433 5                      call terminate;
434 4                  end;
435 3              end;
436 2          if close$file(.fcb) = OFFH then
437 2              do;
438 3                  call print$consoleshuf(.(cr,lf,'Close error on HELP.DAT',
439 3                                              cr,lf,'$'));
440 3                  cnt = close$file(.fcb3);
441 3                  call terminate;
442 2              end;
443 2          if close$file(.fcb3) = OFFH then
444 3              do;
445 3                  call print$consoleshuf(.(cr,lf,'Close error on HELP.HLP',
446 3                                              cr,lf,'$'));
447 2                  call terminate;
448 2                  call print$consoleshuf(.(('HELP.HLP created',cr,lf,'$')));
end create$index;

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

/*****

Extract\$file Procedure

Creates HELP.DAT from HELP.HLP

*****/

449 1 extract\$file;

procedure;

```

450 2      declare (end$index,i,eod) byte;
451 2      declare (count,count2,max$buf,save$size) address;

452 2          call print$console$buf(.(cr,lf,'Extracting data....$'));
453 2          call movef(13,.(0,'HELP  HLP',0),.fcb);
454 2          if open$file(.fcb) = OFFH then
455 2              do;
456 3                  call print$console$buf(.(cr,lf,'Unable to find file HELP.HLP',
                                          cr,lf,'$'));
457 3                  call terminate;
458 3              end;
459 2          call movef(13,.(0,'HELP  DAT',0),.fcb2);
460 2          call delete$file(.fcb2);
461 2          if make$file(.fcb2) = OFFH then
462 2              do;
463 3                  call print$console$buf(.(cr,lf,'Unable to Make HELP.DAT',
                                          cr,lf,'$'));
464 3                  i = close$file(.fcb);
465 3                  call terminate;
466 3              end;
467 2          call set$dma(.sysbuff);
468 2          end$index = 0;
469 2          do while ((i := read$record(.fcb)) = 0) and (not end$index);
470 3              if sysbuff(7).subject(0) = '$' then end$index = true;
471 3          end;
472 2          eod = 0;
473 2          if i <> 0 then eod = true;
474 2          i = write$record(.fcb2);
475 2          save$size = maxb - .memory;
476 2          do while (not eod);
477 3              count = 0;
478 3              max$buf = save$size;
479 3              do while (not eod) and (max$buf > 127);
480 4                  call set$dma(.memory(shl(count,7)));
481 4                  if read$record(.fcb) <> 0 then
482 4                      do;
483 5                          eod = true;
484 5                      end;
485 4                  else
486 4                      do;
487 5                          max$buf = max$buf - 128;
488 5                          if max$buf > 128 then
489 5                              do;
490 6                                  count = count + 1;
491 6                                  end;
492 5                              end;
493 5                          end;
494 4                      do count2 = 0 to count;
495 4                          call set$dma(.memory(shl(count2,7)));
496 4                          if write$record(.fcb2) = OFFH then
497 4                              do;
498 5                                  call print$console$buf(.(cr,lf,'Write error on file HELP.DAT',
499 5                                                          cr,lf,'$'));
500 5                                  i = close$file(.fcb2);
501 5                                  call delete$file(.fcb2);
502 5                                  i = close$file(.fcb);
503 5                                  call terminate;

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

504 5         end;
505 4         end;
506 3         end;
507 2         if close$file(.fcb) = OFFH then
508 2         do;
509 3             call print$console$buf(.(cr,lf,'Unable to Close HELP.HLP',
                    ' cr,lf,'$'));
510 3         end;
511 2         if close$file(.fcb2) = OFFH then
512 2         do;
513 3             call print$console$buf(.(cr,lf,'Unable to Close HELP.DAT',
                    cr,lf,'$'));
514 3             call delete$file(.fcb2);
515 3             call terminate;
516 3         end;
517 2         call print$console$buf(.(cr,lf,'Extraction complete',cr,lf,lf,
                    'HELP.DAT created',cr,lf,'$'));

```

```

518 2     end extract$file;

```

```

/*****

```

Display\$ind Procedure

Displays the available topics

```

*****/

```

```

519 1 display$ind:
    procedure;
520 2     declare (disp$level,i,eod,written) byte;
521 2     declare (offset,index,count) address;
522 2     declare name (14) byte;
523 2     offset,
        written,
        eod = 0;
524 2     disp$level = level + 1;
525 2     if disp$level < 10 then
526 2     do;
527 3         if level = 0 then
528 3         do;
529 4             offset = 0;
530 4             end;
531 3         else
532 3         do;
533 4             offset = gindex;
534 3             end;
535 3             count = 0;
536 2         end;
537 2         else
538 2         do;
539 3             eod = true;
540 3             end;
541 2             index = offset;
542 2             offset = 0;
543 2             do while (not eod);
544 3                 if sysbuff(index).subject(0) = '$' then
545 3                 do;
546 4                     eod = true;
547 3                     end;

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

else
do;
546 3      if sysbuff(index).level = disp$level then
547 4      do;
548 4          if not written then
549 5          do;
550 5              written = true;
551 6              i = page$check(.tcnt);
552 6              if disp$level = 1 then
553 6              do;
554 6                  call print$console$buf(.(cr,'Topics available:$'));
555 7              end;
556 7              else
557 6              do;
558 7                  call print$console$buf(.(cr,'Additional topics ',
                    'available:$'));
559 7              end;
560 6              i = page$check(.tcnt);
561 6              call print$console$buf(.(cr,$'));
562 6          end;
563 5          if (count mod 6) = 0 then
564 5          do;
565 6              i = page$check(.tcnt);
566 6              call print$console$buf(.(cr,$'));
567 6          end;
568 5          do i = 0 to 13;
569 6              name(i) = ' ';
570 6          end;
571 5          name(13) = '$';
572 5          call movef(12,sysbuff(index).subject,.name);
573 5          call print$console$buf(.name);
574 5          count = count + 1;
575 5          end;
576 4          else
577 5          do;
578 5              if sysbuff(index).level < disp$level then eod = true;
579 5          end;
580 4          index = index + 1;
581 4          end;
582 3      end;
583 2      if written then call print$console$buf(.(cr,lf,$'));
584 2      call set$dma(.tbuf);
585 2      end display$ind;

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

/*****

Search\$file Procedure

Searches the index table for the key

*****/

```

587 1      search$file;
588 1          procedure byte;
589 2          declare (eod, error, cnt, found, saved, save$level) byte;
590 2          declare index address;
591 2          eod,
592 2          error,
593 2          found,
594 2          saved,

```

```

index = 0;
591 2 do while(not eod) and (not error);
592 3   if sysbuff(index).subject(0) <> '$' then
593 3     do;
594 4       if sysbuff(index).level = level + 1 then
595 4         do;
596 5           cnt = compare(.com(level).name,.sysbuff(index).subject);
597 5           if cnt = 0 then
598 5             do;
599 6               call movef(12,.sysbuff(index).subject,.com(level).name);
600 6               level = level + 1;
601 6               if (not saved) then
602 6                 do;
603 7                   save$level = level;
604 7                   saved = true;
605 7               end;
606 6               if ((level > 8) or (com(level).name(0) = ' '))
                   or (com(level).name(0) = '[') then
607 6                 do;
608 7                   found = true;
609 7                   eod = true;
610 7                 end;
611 6               else
612 7                 do;
613 7                   index = index + 1;
614 7                   found = 0;
615 6                 end;
616 5               end;
617 6               else
618 6                 do;
619 5                   index = index + 1;
620 4                 end;
621 5               end;
622 5               if saved then
623 6                 do;
624 6                   if save$level < sysbuff(index).level then
625 7                     do;
626 7                       index = index + 1;
627 6                     end;
628 7                     else
629 7                       do;
630 6                         error = true;
631 5                       end;
632 6                       end;
633 5                       else
634 5                         do;
635 4                           index = index + 1;
636 3                         end;
637 4                         end;
638 4                         error = true;
639 3                         end;
end;

```

 COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

640 2      if found then
641 2      do;
642 3          gindex = index + 1;
643 3          call movef(1,.sysbuff(index).record,.fcb(33));
644 3          call movef(1,.sysbuff(index).record+1,.fcb(34));
645 3          fcb(35) = 0;
646 3          offset = sysbuff(index).rec$offset;
647 3          level = sysbuff(index).level;
648 3      end;
649 2      return error;
650 2  end search$file;

```

```

/*****
      Token Display Procedure

```

```

      Displays the Parsed Tokens

```

```

*****/
651 1  display$tokens:
      procedure (not$tokens);
652 2      declare (token$cnt1, token$cnt2, not$tokens) byte;
653 2      token$cnt1 = 0;
654 2      do while (token$cnt1 < not$tokens) and (not eod);
655 3          eod = page$check(.tcnt);
656 3          if (not eod) then
657 3          do;
658 4              do token$cnt2 = 0 to token$cnt1;
659 5                  call print$console$buf(.( ' $' ));
660 5              end;
661 4              call print$console$buf(.(com(token$cnt1).name));
662 4              token$cnt1 = token$cnt1 + 1;
663 4          end;
664 3      end;
665 2  end display$tokens;

```

```

/*****
      Print Procedure

```

```

      Displays the Help text

```

```

*****/
666 1  print:
      procedure;
667 2      declare (i,ii,char,eod2) byte;
668 2      declare temp(3) byte;
669 2      call write$console(cr);
670 2      call display$tokens(level);
671 2      if (not eod) then eod = page$check(.tcnt);
673 2      if (not eod) then
674 2      do;
675 3          if read$rand(.fcb) <> 0 then
676 3          do;
677 4              call print$console$buf(.(cr,lf,'HELP.HLP read error',cr,lf,'$'));
678 4              offset =close$file(.fcb);
679 4              call terminate;
680 4          end;
681 3          else
682 4          do;
              eod2 = 0;

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____


```

683 4      do while ((not eod2) and (not eod)) and (read$record (.fcb) = 0);
684 5          i = offset - 1;
685 5      do while (((i:=i+1) <= 127) and (not eod2));
686 6          if (char := tbuff(i)) = controltz then eod = true;
688 6          ii = 0;
689 6          do while((not eod2) and (not eod)) and
              ((ii < 3) and (tbuff(i) = slash));
690 7              ii = ii + 1;
691 7              i = inc(i);
692 7              temp(ii-1) = tbuff(i);
693 7          end;
694 6          if ii = 3 then eod2 = true; else temp(ii) = '$';
697 6          if ((not eod) and (not eod2)) then
698 6              do;
699 7                  if (char = lf) and (not print$mode) then
700 7                      do;
701 8                      eod = page$check(.tcnt);
702 8                      end;
703 7                      else
704 8                      do;
705 8                          call write$console (char);
706 7                          end;
707 7                          if ii > 0 then call print$console$buf(.temp);
708 7                          ii = 0;
709 7                      end;
710 6                  end;
711 5                  offset = 0;
712 5              end;
713 4              end;
714 3          end;
715 2          eod = 0;
716 2      end print;

```

```

/*****
Prompt Procedure

```

Prompts for input from the user

```

*****/
1717 1 prompt;
1718 2     procedure byte;
1719 2         declare temp byte;
1720 2         call movef(1, (128), .tbuff-1);
1721 2         temp = page$check(.tcnt);
1722 2         call print$console$buf(.cr, 'HELP> $');
1723 2         call read$console$buf(.tbuff-1);
1724 2         tbuff(tbuff(0)+1) = 0;
1725 2         tcnt = -1;
1726 2         temp = parse;
1727 2         if (temp <> 0) and (not print$mode)
1728 2             then call print$console$buf(.clear$screen);
1729 2         return temp;
1730 2     end prompt;

```

```

/*****
Main Program
*****/

```

CR/m-86 v.1.1 (vol. II)

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # C81-162

```

730 1  declare lastdseg$byte byte
      initial (0);

731 1  pla:
      do;
732 2      eod,
          tcnt = 0;
733 2      cnt = parse;
734 2      if create$mode then
735 2          do;
736 3          call create$index;
737 3          end;
          else
738 2      if extract$mode then
739 2          do;
740 3          call extract$file;
741 3          end;
          else
742 2      do;
743 3          call movef(13,.(0,'HELP ',0A0H,' HLP',0),.fcb); /* open read/only */
744 3          if open$file (.fcb) <> OFFH then
745 3              do;
746 4              fcb(32) = 0;
747 4              call init;
748 4              call print$console$buf(.clear$screen);
749 4              if cnt = 0 then
750 4                  do;
751 5                  level = 0;
752 5                  call print$console$buf(. (cr,lf,'HELP UTILITY V1.0',cr,lf,lf,
                                          'At 'HELP>' enter ',
                                          'topic {,subtopic}...',cr,lf,lf,
                                          'EXAMPLE: HELP> DIR EXAMPLES',
                                          cr,lf,'$'));
753 5                  tcnt = 2;
754 5                  call display$ind;
755 5                  cnt = prompt; /* Prompt for user input */
756 5                  end;
757 4                  do while cnt <> 0; /* If user didn't hit a return do */
758 5                      level = 0;
759 5                      if search$file <> OFFH then
760 5                          do;
761 6                          call print;
762 6                          if compare(.com(0).name,.( 'HELP      ')) = 0 then
763 6                              do;
764 7                              level = 0;
765 7                              end;
766 6                          end;
                          else
767 5                      do;
768 6                          eod = page$check(.tcnt);
769 6                          call write$console(cr);
770 6                          if (not eod) then
771 6                              do;
772 7                              eod = page$check(.tcnt);
773 7                              if (not eod) then

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950
 SER. # _____

```

774 7          do;
775 8              call print$console$buf(,('Topic:$'));
776 8              eod = page$check(.tcnt);
777 8              call write$console(cr);
778 8              call display$tokens(cnt);
779 8              eod = page$check(.tcnt);
780 8              call write$console(cr);
781 8              eod = page$check(.tcnt);
782 8              call write$console(cr);
783 8              call print$console$buf(,('Not found$'));
784 8              eod = page$check(.tcnt);
785 8              call write$console(cr);
786 8          end;
787 7      end;
788 6      level = 0;
789 6      end;
790 5      if (not eod) then call display$ind;
792 5      cnt = prompt; /* Prompt for user input */
793 5      end;
794 4      offset = close$file(.fcb);
795 4      end;
796 3      else
797 4      do;
          call print$console$buf(,('lf,cr,'No .HLP file on the default ',
                                  'drive',lf,cr,'$'));
798 4      end;
799 3      end;
800 2      end;
801 1      call terminate;
802 1      end help;

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

CROSS-REFERENCE LISTINGDEFN ADDR SIZE NAME, ATTRIBUTES, AND REFERENCES

205	012EH	1	BRACKET.	BYTE	220	237	248											
44	0004H	2	BUFFADR.	WORD	PARAMETER	AUTOMATIC												
40	0004H	2	BUFFADR.	WORD	PARAMETER	AUTOMATIC												
174	0002H	2	BUFSIZE.	WORD		176	177											
667	0176H	1	CHAR	BYTE		686	699	704										
36	0004H	1	CHAR	BYTE	PARAMETER	AUTOMATIC												
32	010CH	26	CLEARSCREEN. . . .	BYTE	ARRAY(26)	INITIAL												
60	01DAH	16	CLOSEFILE.	PROCEDURE	BYTE	STACK=000AH												
					356	379	381	395	397	399	429	431	436	439				
					442	464	500	502	507	511	678	794						
205	012BH	1	CNT.	BYTE		219	221	222	225	226	228	231	233					
					241	246	252											
273	0130H	1	CNT.	BYTE		320	323	325	329	330	333	336	339					
					356	379	381	385	395	397	399	404	439					
538	016EH	1	CNT.	BYTE		596	597											
24	004BH	1	CNT.	BYTE		733	749	755	757	778	792							
26	0059H	165	CON.	STRUCTURE	ARRAY(11)													
					261	263	268	273	278	596	599	606	661	762				
97	026FH	93	COMPARE.	PROCEDURE	BYTE	STACK=0008H												
18			CONTROLZ.	LITERALLY		313	686											
274	000AH	2	COUNT.	WORD		359	360	387	388	390	391	408	411					
					420	424												
451	0012H	2	COUNT.	WORD		479	482	491	495									
521	001EH	2	COUNT.	WORD		534	563	574										
294	000CH	2	COUNT2.	WORD		424	425											
451	0014H	2	COUNT2.	WORD		495	496											
19			CR	LITERALLY		32	144	149	156	185	199	210						
					296	306	330	355	378	394	428	438	444	447				
					452	456	463	499	509	513	517	555	558	561				
					566	584	669	677	721	752	769	777	780	782				
					785	797												
292	0668H	1211	CREATEINDEX. . . .	PROCEDURE	STACK=0014H													
11	0023H	1	CREATEHODE.	BYTE		256	265	734										
52	01BAH	16	DELETEFILE.	PROCEDURE	STACK=000AH													
					430	460	501	514										
91	0004H	2	DESTADDR.	WORD	PARAMETER	AUTOMATIC												
48	01A7H	19	DIRECTCONIO. . . .	PROCEDURE	BYTE	STACK=000AH												
519	0CD1H	350	DISPLAYIND.	PROCEDURE	STACK=0014H													
651	0F9CH	104	DISPLAYTOKENS. . .	PROCEDURE	STACK=0016H													
520	015AH	1	DISPLEVEL.	BYTE		524	525	547	553	577								
30	0004H	2	DMAADDRESS.	WORD	PARAMETER	AUTOMATIC												
450	0157H	1	ENDINDEX.	BYTE		468	469	471										
175	0129H	1	ENDINDEX.	BYTE		178	180	189	190	197								
588	016CH	1	EOD.	BYTE		590	591	609										
12	0026H	1	EOD.	BYTE		129	654	655	656	671	672	673	683					
					687	689	697	701	715	732	768	770	772	773				
					776	779	781	784	790									
520	015CH	1	EOD.	BYTE		523	537	541	544	578								
450	0159H	1	EOD.	BYTE		473	475	478	481	485								

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

 45 46
 41 42 P.O. BOX 579
 PACIFIC GROVE, CA 93950

375 38=

727 748

152 184 200

273	012FH	1	EOD.	BYTE	309	310	312	315	321	325	330	341
					353	406	407	410	414			
667	0177H	1	EOD2	BYTE	682	683	685	689	695	697		
588	016DH	1	ERROR.	BYTE	590	591	628	637	649			
449	0323H	430	EXTRACTFILE. . . .	PROCEDURE	STACK=000EH				740			
11	0024H	1	EXTRACTNODE. . . .	BYTE	256	270	738					
13	0000H	13	FC2.	BYTE ARRAY(13)	EXTERNAL(2)				124	152	182	
					184	200	303	304	310	343	344	345
					386	399	407	404	405	412	431	436
					464	469	483	502	507	643	644	645
					683	743	744	746	794			
16	0000H	1	FCB16.	BYTE ARRAY(1)	EXTERNAL(4)							
14	0027H	36	FCB2	BYTE ARRAY(36)		379	380	384	385	397	398	
					459	460	461	476	497	500	501	511
295	0133H	36	FCB3	BYTE ARRAY(36)		374	375	376	386	392	395	
					396	426	429	430	439	442		
64	0004H	2	FCBADDRESS	WORD	PARAMETER	AUTOMATIC						
60	0004H	2	FCBADDRESS	WORD	PARAMETER	AUTOMATIC						
56	0004H	2	FCBADDRESS	WORD	PARAMETER	AUTOMATIC						
72	0004H	2	FCBADDRESS	WORD	PARAMETER	AUTOMATIC						
76	0004H	2	FCBADDRESS	WORD	PARAMETER	AUTOMATIC						
68	0004H	2	FCBADDRESS	WORD	PARAMETER	AUTOMATIC						
84	0004H	2	FCBADDRESS	WORD	PARAMETER	AUTOMATIC						
52	0004H	2	FCBADDRESS	WORD	PARAMETER	AUTOMATIC						
588	016FH	1	FOUND.	BYTE	590	608	613	640				
48	0004H	1	FUNC	BYTE	PARAMETER	AUTOMATIC						
7	0000H	1	FUNC	BYTE	PARAMETER		8					
3	0000H	1	FUNC	BYTE	PARAMETER		4					
30	0000H	2	GINDEX	WORD	532	642						
1	0002H	355	HELP	PROCEDURE	STACK=001CH							
101	0127H	1	I.	BYTE	102	103	104	106	113	115		
667	0174H	1	I.	BYTE	684	685	686	689	691	692		
520	015BH	1	I.	BYTE	552	560	565	568	569			
450	0153H	1	I.	BYTE	464	469	474	476	500	502		
293	0131H	1	I.	BYTE	297	298	311	312	313	318	321	322
					327	328	330	331	333	336	338	347
					431							
205	012CH	1	I.	BYTE	224	225	228	231	234	259	260	261
					263	268	273	278	285	287		
667	0175H	1	II	BYTE	688	689	690	692	694	696	706	708
119	02CCH	42	INC.	PROCEDURE	BYTE STACK=0010H				322	328	338	
					367	691						
119	0004H	1	INCI	BYTE	PARAMETER	AUTOMATIC			120	121	122	126
					130	133						
24	004CH	1	INDEX.	BYTE								
521	001CH	2	INDEX.	WORD	539	542	547	572	577	580		
589	0020H	2	INDEX.	WORD	590	592	594	596	599	612	617	623
					625	632	642	643	644	646	647	
274	0008H	2	INDEX.	WORD	300	327	333	336	344	345	346	347
					348	349	359	387				
205	012AH	1	INDEX.	BYTE	206	209	210	212	213	214	215	216
					218	225	228	231	236	237	243	244
					257	266	271	276	280	283	290	
7	0000H	2	INFO	WORD	PARAMETER		9					
3	0000H	2	INFO	WORD	PARAMETER		5					
173	037EH	194	INIT	PROCEDURE	STACK=000EH				747			
174	0006H	2	INITI.	WORD	179	181	184	188	193	200		

65 66
61 62
57 58
73 74
77 78
69 70
85 86
53 54
49 50
SER. #

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 570
PACIFIC GROVE, CA 93950

730	017CH	1	LASTDSEGBYTE . . .	BYTE INITIAL								
29	010AH	1	LEVEL.	BYTE	524	527	594	596	599	600	603	606
					647	670	751	758	764	788		
27	000FH	1	LEVEL.	BYTE MEMBER(SYSBUFF)				327	547	577	594	623
					647							
20			LF	LITERALLY		32	144	164	169	185	199	296
					306	355	378	394	428	438	444	452
					463	499	509	513	517	584	677	797
137	0000H	1	LINECNT.	BYTE BASED(LINECNTADDR)					142	145	146	147
					150	152	156	160	168			
135	0004H	2	LINECNTADDR. . . .	WORD PARAMETER AUTOMATIC					136	137	142	145
					146	147	150	152	156	160	168	
72	020AH	16	MAKEFILE	PROCEDURE BYTE STACK=000AH					376	461		
15	0000H	2	MAXB	WORD EXTERNAL(3)				176	30	477		
451	0016H	2	MAXBUF	WORD	480	481	488	489				
294	000EH	2	MAXBUF	WORD	302	352	353	409	410	417	418	
174	0004H	2	MAXBUF	WORD	177	180	192	197				
	0000H		MEMORY	BYTE ARRAY(0)			27	176	301	391	411	425
					477	482	496					
3	0000H		MON1	PROCEDURE EXTERNAL(0) STACK=0000H						38	42	
					46	54	82	86	89			
7	0000H		MON2	PROCEDURE BYTE EXTERNAL(1) STACK=0000H								34
					50	58	62	66	70	74	78	
			MOVE	BUILTIN				94				
91	0258H	23	MOVEF.	PROCEDURE STACK=000AH					210	298	303	344
					345	360	374	384	403	453	459	572
					644	719	743					643
91	0008H	1	MVCNT.	BYTE PARAMETER AUTOMATIC					93	94		
28	00FEH	12	NAME	BYTE ARRAY(12)								
522	015EH	14	NAME	BYTE ARRAY(14)			569	571	572	573		
26	0000H	15	NAME	BYTE ARRAY(15) MEMBER(COM)						210	228	231
					243	245	261	263	268	273	278	596
					661	762						606
651	0004H	1	NOTOKENS	BYTE PARAMETER AUTOMATIC					652	654		
521	001AH	2	OFFSET	WORD	523	529	532	539	540			
12	0025H	1	OFFSET	BYTE	646	678	684	711	794			
56	01CAH	16	OPENFILE	PROCEDURE BYTE STACK=000AH						304	454	744
135	02F6H	136	PAGECHECK.	PROCEDURE BYTE STACK=0010H						552	560	565
					655	672	701	720	768	772	776	779
											781	784
204	0440H	552	PARSE.	PROCEDURE BYTE STACK=000EH						725	733	
2	0010H		PLN.	LABEL PUBLIC				731				
666	1004H	382	PRINT.	PROCEDURE STACK=001AH					761			
40	0187H	16	PRINTCONSOLEBUF. .	PROCEDURE STACK=000AH					144	149	185	199
					296	306	355	378	394	428	438	444
					456	463	499	509	513	517	555	558
					573	584	659	661	677	707	721	727
					775	783	797				748	752
11	0022H	1	PRINTMODE.	BYTE	140	256	275	699	726			
717	1182H	93	PROMPT	PROCEDURE BYTE STACK=0014H						755	792	
138	0128H	1	QUIT	BYTE	139	158	171					
33	0165H	15	READCONSOLE. . . .	PROCEDURE BYTE STACK=0008H								
44	0197H	16	READCONSOLEBUF. .	PROCEDURE STACK=000AH					722			
76	021AH	16	READRAND	PROCEDURE BYTE STACK=000AH						385	404	675
64	01E3H	16	READRECORD	PROCEDURE BYTE STACK=000AH						124	182	310
					412	469	483	683				
293	0132H	1	RECCNT	BYTE	300	351	373	388	390			

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 570
PACIFIC GROVE, CA 93950

27	000EH	1	RECOFFSET.	BYTE MEMBER(SYSBUFF)	347	646			
27	000CH	2	RECORD	WORD MEMBER(SYSBUFF)	344	345	346	388	643
				644					
101	0126H	1	RESULT	BYTE	102	108	111	117	
588	0170H	1	SAVED.	BYTE	590	601	604	621	
588	0171H	1	SAVELEVEL.	BYTE	603	623			
451	0013H	2	SAVESIZE	WORD	477	480			
294	0010H	2	SAVESIZE	WORD	301	302	409		
587	0E2FH	345	SEARCHFILE	PROCEDURE BYTE STACK=000EH					
80	022AH	16	SETDMA	PROCEDURE STACK=000AH	411	425	467	482	496
					585				
84	023AH	16	SETRANDREC	PROCEDURE STACK=000AH					
			SHL.	BUILTIN	391	411	425	482	496
22			SLASH.	LITERALLY	318	321	689		
91	0003H	2	SOURCEADDR	WORD PARAMETER AUTOMATIC					
205	012DH	1	STOP	BYTE	220	221	239		
77	0006H	2	STR1ADDR	WORD PARAMETER AUTOMATIC				98	99
				106				103	104
77	0004H	2	STR2ADDR	WORD PARAMETER AUTOMATIC				98	100
99	0000H	12	STRING1.	BYTE BASED(STR1ADDR) ARRAY(12)				103	104
100	0000H	12	STRING2.	BYTE BASED(STR2ADDR) ARRAY(12)				104	106
25	004DH	12	SUB.	BYTE ARRAY(12)					
27	0000H	12	SUBJECT.	BYTE ARRAY(12) MEMBER(SYSBUFF)				181	188
				333	336	360	470	542	572
				592	596	599			
27	0000H	128	SYSBUFF.	STRUCTURE ARRAY(8) AT				181	188
				333	336	344	345	346	347
				360	372	388	467		
				470	542	547	572	577	592
				594	596	599	623		
				643	644	646	647		
21			TAB.	LITERALLY	213				
17	0000H	128	TBUFF.	BYTE ARRAY(128) EXTERNAL(5)				196	207
				213	214	215	216	221	222
				225	226	228	231		
				241	313	318	321	327	330
				331	333	336	585		
				686	689	692	719	722	723
31	010BH	1	TCNT	BYTE	552	560	565	655	672
				732	753	768	772	776	779
								781	784
718	017BH	1	TEMP	BYTE	720	725	726	728	
668	0178H	3	TEMP	BYTE ARRAY(3)				692	696
								707	
88	024AH	14	TERMINATE.	PROCEDURE STACK=0008H				153	186
				357	382	400	432	440	445
				457	465	503	515		
				679	801				
652	0172H	1	TOKENCNT1.	BYTE	653	654	658	661	662
652	0173H	1	TOKENCNT2.	BYTE	658				
23			TRUE	LITERALLY		129	158	189	239
				275	315	414	471	475	485
				537	544	551	578		
				604	608	609	628	637	687
				695					
36	0174H	19	WRITECONSOLE . . .	PROCEDURE STACK=000AH				164	169
				769	777	780	782	785	
68	01FAH	16	WRITERECORD. . . .	PROCEDURE BYTE STACK=000AH				392	426
				497				476	
520	015DH	1	WRITTEN.	BYTE	523	549	551	583	

COPYRIGHT 1981, 1982
DIGITAL RESEARCH
P.O. BOX 100
PACIFIC GROVE, CA 93950
SER ==

MODULE INFORMATION:

CODE AREA SIZE = 11DFH 4575D

CONSTANT AREA SIZE = 03FAH 1018D

VARIABLE AREA SIZE = 017DH 331D
MAXIMUM STACK SIZE = 001CH 28D
1050 LINES READ
0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

STR. # _____